Informatik
Hauptcampus

HOCHSCHULE TRIER

# Prozedurale Generierung von Baumstrukturen innerhalb der Unreal Engine 4

Procedural generation of tree-like structures in the Unreal Engine 4

David Liebemann

Bachelor-Abschlussarbeit

Betreuer:
Prof. Dr. Christof Rezk-Salama

Trier, 26.02.2017

# Kurzfassung

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1

# Einleitung und Problemstellung

## 1.1 Prozedurale Generierung

Insbesondere Vegetation - warum, was sind die Schwierigkeiten, wenn man es nicht prozedural generiert.

## 1.2 Bisherige Arbeiten

## 1.3 Unreal Engine 4

Allgemeine Erklärung - was ist die Unreal Engine 4, was stellt sie zur Verfügung, wie entwickelt man dafür? Insbesondere Eingabe von Parametern über den Editor –> wichtig für Implementierung.

## 1.4 Ansatz

Verwendete Ansätze: L-Systeme mit Turtle-Graphik und Space Colonization Algorithmus. Kleine Einleitung zu beiden Ansätzen, bisherige Verwendung.

Alle Längenangaben entsprechen der in der Unreal Engine 4 verwendeten Längeneinheit *cm*.

# 2

# Lindenmayer-Systeme

TODO: Vorstellung der verwendeten Konzepte. Sämtliche Visualisierungen von L-Systemen wurden mithilfe des Unreal Engine 4 Projekts erstellt.

## 2.1 Kontextfreie Grammatik

Bei L-Systemen handelt es sich um, auf Zeichenketten basierende, Ersetzungssysteme. Es werden komplexe Objekte beschrieben, indem Teile der Zeichenkette durch andere Zeichen oder Zeichenketten ersetzt werden. Die Beschreibung dieser Ersetzungen findet mittels festgelegter Produktionsregeln statt. [PL04, S.2]

Eine formale Definition eines auf Zeichenketten arbeitenden Ersetzungssystems wird durch kontextfreie Grammatiken gegeben:

**Kontextfreie Grammatik: 2.1** *Eine kontextfreie Grammatik G ist ein Tupel* $G = (V, N, P, \omega)$ *bestehend aus:*

| | |
|---|---|
| $\boldsymbol{V}$ | *Einer nichtleeren, endlichen Menge von Buchstaben (Alphabet).* |
| $\boldsymbol{N}$ | *Einer endlichen Menge von Variablen.* |
| $\boldsymbol{P}$ | *Einer endlichen Menge von Produktionsregeln in der Form* $P : A \to \alpha$ *mit* $A \in N$ *und* $\alpha \in (V \cup N)^*$. |
| $\boldsymbol{\omega \in N}$ | *Dem Axiom, Startsymbol der Grammatik.* |

*[Sch14, S.343]*

Die Menge $V^*$ ist die Menge aller Wörter über $V$ – die Menge aller Wörter, die aus dem Alphabet $V$ gebildet werden können. [Sch14, S.70]

Eine Grammatik wird als kontextfrei bezeichnet, wenn beispielsweise die Produktionsregel $A \to \alpha$ angewendet werden kann, ohne die $A$ umgebenden Buchstaben – seinen Kontext – beachten zu müssen. [Sch14, S.343]

Eine Grammatik wird als deterministisch bezeichnet, wenn es genau eine Produktionsregel $r \in P$ für jede Variable $A \in V$ gibt, sodass $r : A \to \alpha$, $\alpha \in (V \cup N)^*$.

Das bedeutet, dass die Ersetzung einer Variable eindeutig durch eine einzige Regel beschrieben wird. [STN16, S.75]

Die Anwendung der Produktionsregeln findet meist sequentiell statt – die Zeichenkette wird von links nach rechts durchlaufen und Ersetzungen werden direkt auf die untersuchte Zeichenkette angewendet. [STN16, S.75]

## 2.2 D0L-Systeme

Diese Arbeit beschränkt sich auf die Behandlung deterministischer, kontextfreier L-Systeme, auch D0L-Systeme genannt. Diese besitzen die Eigenschaften einer deterministischen und kontextfreien Grammatik, Produktionsregeln werden jedoch parallel und gleichzeitig auf alle Buchstaben des untersuchten Wortes angewendet. Dieses Vorgehen soll die Zellteilung in mehrzelligen Organismen simulieren und ist somit an biologische Vorgänge angelehnt. [PL04, S. 3]

Ein D0L-System kann wie folgt definiert werden:

**D0L-System: 2.1** *Ein D0L-System ist ein Tupel $G = (V, P, \omega)$, bestehend aus:*

***V*** *Einem nichtleeren, endlichen Alphabet.*

***P*** *Einer endlichen Menge von Produktionsregeln in der Form $P : a \to b$ mit $a \in V$ und $b \in V^*$. $a$ wird als Vorgänger, $b$ als Nachfolger bezeichnet. Ist für einen Buchstaben $x \in V$ keine explizite Produktionsregel angegeben, wird die Identitätsproduktion $P : x \to x$ angenommen – der Buchstabe wird durch sich selbst ersetzt.*

***$\omega \in V^+$*** *Dem Axiom, Startsymbol der Grammatik.*

*[PL04, S.4]*

Die Menge $V^+$ ist die Menge aller nichtleeren Wörter über $V$.

Die Ableitung eines Wortes entspricht der Ersetzung aller Buchstaben anhand der Produktionsregeln. Ein Wort kann mehrmals abgeleitet werden.

**Ableitung: 2.1** *Gegeben sei ein Wort $w = a_1...a_m$ mit $w \in V^*$ und $a_i \in V^*$. Das Wort $v = b_1...b_m$ mit $v \in V^*$ und $b_i \in V^*$ ist die Ableitung von $w$ wenn für alle $i = 1...m$ eine Produktionsregel $a_i \to b_i$ existiert. Die Ableitung wird als $w \Rightarrow v$ notiert.*
*Das Wort $w_n$ ist die n-te Ableitung des Wortes $w_0$ wenn eine Folge von Wörtern $w_0, w_1, ..., w_n$ mit Ableitungen $w_0 \Rightarrow w_1 \Rightarrow ... \Rightarrow w_n$ existiert. [PL04, S.4]*

Beispiel: Das Wachstum der Blaualgen-Gattung „Anabaena“ kann durch ein L-System simuliert werden. Die Buchstaben $a$ und $b$ beschreiben die Größe und Teilungsbereitschaft einer Algenzelle, während die Indizes $l$ und $r$ die Polarität einer Zelle darstellen. Es gelten folgende Produktionsregeln:

$$\begin{aligned} p_1 &: a_r \rightarrow a_l b_r \\ p_2 &: a_l \rightarrow b_l a_r \\ p_3 &: b_r \rightarrow \ a_r \\ p_4 &: b_l \rightarrow \ a_l \end{aligned} \tag{2.1}$$

Die Entwicklung einer Anfangszelle $a_r$ (Axiom $\omega : a_r$) läuft daraufhin wie in Abbildung 2.1 dargestellt ab.

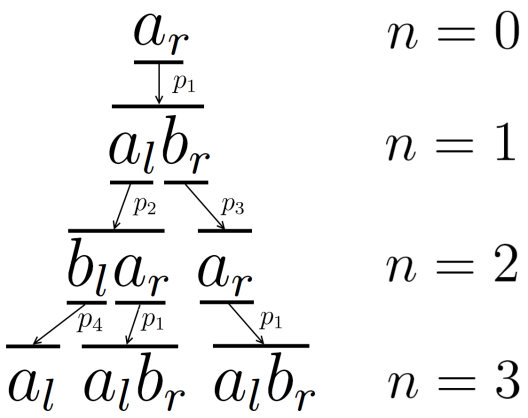



Abb. 2.1: Die n-fache Ableitung der Anfangszelle $a_r$ anhand der Produktionsregeln $p_1 ... p_4$ aus Gleichung 2.1. Eigene Abbildung auf Basis von [PL04, S.4].

### 2.2.1 Parametrische L-Systeme

Parametrische L-Systeme stellen eine Erweiterung der D0L-Systeme dar. Die Buchstaben eines verwendeten Alphabets $V$ werden um zugeordnete Parameter aus der Menge der reelen Zahlen ergänzt. Ein solches parametrisches Wort $V \times \Re^*$ besteht aus einem Zeichen $A \in V$ und Parametern $a_1, ..., a_n \in \Re$ und wird als $A(a_1, ..., a_n)$ dargestellt. Ein parametrisches Wort ohne Parameter mit dem Zeichen $A \in V$ wird schlicht als $A$ dargestellt. [PL04, S.41]

Während in der Definition von Modulen numerische Konstanten verwendet werden, werden bei der Angabe eines L-Systems Variablen verwendet. Ist $\Sigma$ eine Menge von Variablen, dann ist $E(\Sigma)$ ein arithmetischer Ausdruck, in dem Variablen, Konstanten und arithmetische Operatoren auf eine zulässige Weise kombiniert werden. [PL04, S.41]

**Parametrisches L-System: 2.2.1** *Ein Parametrisches L-System ist ein Tupel* $G = (V, \Sigma, P, \omega)$*, bestehend aus:*

***V*** *Einem nichtleeren, endlichen Alphabet.*

***Σ*** *Einer Menge der Variablen.*

$\boldsymbol{P}$ *Einer endlichen Menge von Produktionsregeln* $P : (V \times \Sigma^*) \rightarrow (V \times E(\Sigma))^*$

$\boldsymbol{\omega \in M^+}$ *mit* $M = (V \times \Re^*)$ *– einem Axiom in Form eines nichtleeren, parametrischen Wortes.*

*[PL04, S.41]*

Eine Produktion kann auf ein parametrisches Wort angewendet werden wenn das Zeichen, das dem Wort vorausgeht, und die Anzahl der Parameter mit dem Zeichen und der Anzahl der Parameter im Vorgänger der Produktionsregel übereinstimmen. [PL04, S.42]

Beispiel: Gegeben sei folgendes, parametrisches L-System:

$$\begin{aligned} \omega &: A(1,1) \\ p_1 &: A(x,y) \rightarrow A(x+1, y*2)\ B(y) \\ p_2 &: B(x) \quad \rightarrow B(x+1)\ C \end{aligned} \tag{2.2}$$

Das Alphabet $V$ und die Menge der Variablen $\Sigma$ gehen implizit aus der Angabe der Produktionsregeln hervor und werden in zukünftigen L-System-Gleichungen nicht angegeben. Die Entwicklung des L-Systems läuft anhand Abbildung 2.2 ab.

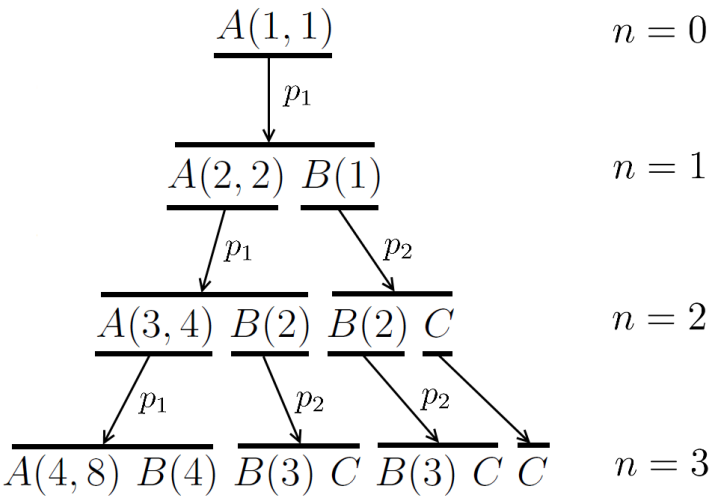



Abb. 2.2: Die n-fache Ableitung des Axioms $A(1,1)$ anhand der Produktionsregeln aus Gleichung 2.2. Die Ersetzung des Zeichens $C$ erfolgt anhand der impliziten Identitätsproduktion. Eigene Abbildung.

## 2.3 Grafische Interpretation von L-Systemen

Um die Ergebnisse von L-Systemen in Form von dreidimensionalen Objekten zu visualisieren, muss eine grafische Interpretation der resultierenden Zeichenket-

ten festgelegt werden. Im Folgenden wird die verwendete Visualisierungsmethode, Turtle-Interpretation, vorgestellt.

### 2.3.1 Turtle-Interpretation

Die Turtle-Interpretation im zweidimensionalen Raum entspricht der Vorstellung einer Turtle[1] auf einem Blatt Papier. Die Turtle besitzt eine Position $\overrightarrow{p}$ sowie einen Einheitsvektor $\overrightarrow{H}$ in kartesischen Koordinaten. $\overrightarrow{p}$ beschreibt die Position der Turtle auf der Ebene und $\overrightarrow{H}$ entspricht der Blickrichtung (Heading) der Turtle. Der Zustand $(\overrightarrow{p}, \overrightarrow{H})$ einer Turtle wird somit vollständig durch die Position und Blickrichtung definiert. [GSJ04, S.2] Die Turtle kann drei Aktionen durchführen, welche durch die folgenden Symbole dargestellt werden:

***F(l)*** Die Turtle bewegt sich um $l > 0$ in die Richtung der aktuellen Blickrichtung. Die neue Position ist $\overrightarrow{p_{neu}}$ mit:

$$\overrightarrow{p_{neu}} = \overrightarrow{p} + \overrightarrow{H} * l \tag{2.3}$$

Zwischen der alten Position $\overrightarrow{p}$ und der neuen Position $\overrightarrow{p_{neu}}$ wird eine Linie gezeichnet.

***+(d)*** Die Turtle dreht sich um den Winkel $d$ nach links. Die neue Blickrichtung ist $\overrightarrow{H'}$ mit:

$$\overrightarrow{H'} = \begin{pmatrix} cos(d) & -sin(d) \\ sin(d) & cos(d) \end{pmatrix} * \overrightarrow{H} \tag{2.4}$$

***−(d)*** Die Turtle dreht sich um den Winkel $d$ nach rechts. Die neue Blickrichtung ist $\overrightarrow{H'}$ mit:

$$\overrightarrow{H'} = \begin{pmatrix} cos(d) & sin(d) \\ -sin(d) & cos(d) \end{pmatrix} * \overrightarrow{H} \tag{2.5}$$

[GSJ04, S.4,46] [PL04, S.7] Die Symbole „+“ und „-“ werden sowohl im Alphabet eines L-Systems als auch bei arithmetischen Operationen in Parameterangaben verwendet, ihre Bedeutung ist abhängig vom Kontext, in dem sie angewendet werden. [PL04, S.46]

Die grafische Turtle-Interpretation einer Zeichenkette, die durch ein L-System zurückgegeben wird, sind somit die Linien, die auf Grundlage der definierten Symbole gezeichnet werden.

Beispiel: Mithilfe von L-Systemen und einer Turtle-Interpretation können Fraktale, in diesem Beispiel sogenannte Koch-Kurven, visualisiert werden. Diese Kurven bestehen aus einem Initiator – einer einfachen, zweidimensionalen Form – und einem Generator, der einem offenen Polygonzug entspricht. In jedem Ableitungsschritt, angefangen bei dem Initiator, wird jede gerade Linie durch den Generator ersetzt. [Man83, S.39]

[1] „Turtle“ – englisch für „Schildkröte“.

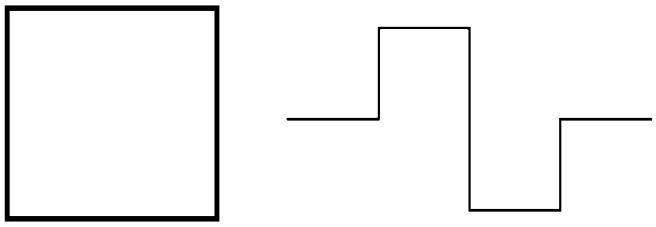


Abb. 2.3: Links: Initiator der Koch-Kurve in Form eines einfachen Quadrats. Rechts: Generator der Koch-Kurve in Form eines offenen Polygonzugs. Eigene Abbildung.

Dieses Verhalten kann nun auf ein L-System abgebildet werden: Der Initiator entspricht dem Axiom und der Generator einer Produktionsregel des L-Systems. Der in Abbildung 2.3 dargestellte Initiator und Generator entsprechen der Turtle-Interpretation des folgenden L-Systems:

$$\begin{aligned} \omega &: F - F - F - F \\ p_1 &: F \rightarrow F + F - F - FF + F + F - F \end{aligned} \tag{2.6}$$

Für eine bessere Übersicht wurde die Angabe der Parameter weggelassen. Die Turtle interpretiert $F$ als $F(l)$, $-$ als $-(d)$ und $+$ als $+(d)$ mit festgelegter Strichlänge $l$ und Drehwinkel $d$. Die Entwicklung des L-Systems läuft, als Turtle-Interpretation visualisiert, wie in Abbildung 2.4 ab.

### 2.3.2 Verzweigte L-Systeme

Die bisherigen Definitionen von L-Systemen und die korrespondierende Turtle-Interpretation lässt lediglich die Bildung von Grafiken zu, die einen einzelnen, zusammenhängenden Polygonzug erlauben. Um L-Systeme zu bilden, deren Visualisierungen Baumstrukturen ähneln muss die bisherige Turtle-Interpretation um die Möglichkeit erweitert werden, Verzweigungen zu verarbeiten. [PL04, S.24]

Folgende Operationen werden eingeführt:

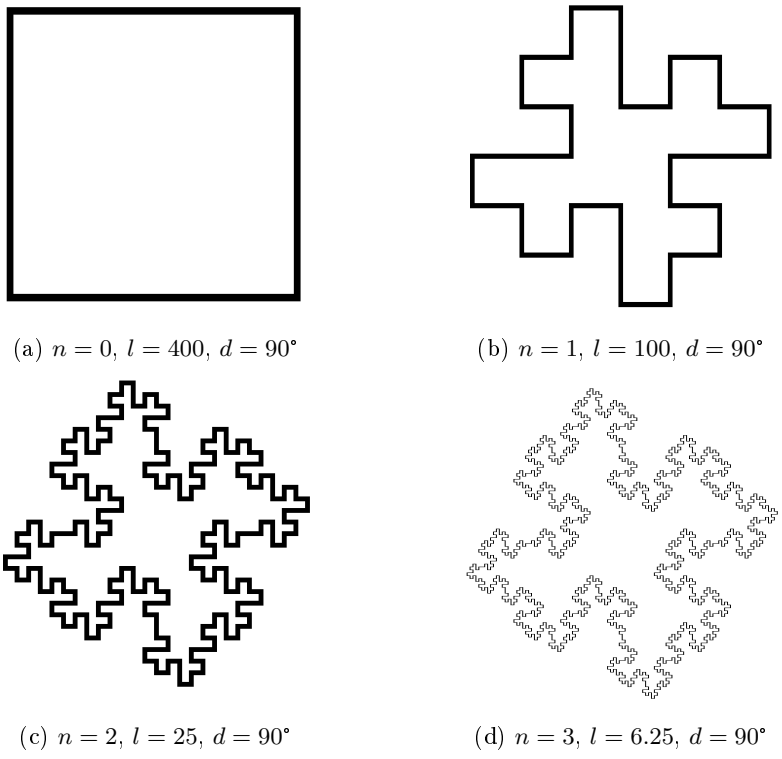
**[** Der aktuelle Zustand der Turtle in Form ihrer Position und Rotation wird auf einem Kellerspeicher abgelegt.

**]** Der oberste Zustand der Turtle wird vom Kellerspeicher genommen. Die aktuelle Position und Rotation der Turtle wird auf die im Zustand gespeicherte Position und Rotation gesetzt. Es wird keine Linie zwischen der alten und neuen Position gezeichnet.

[PL04, S.24]

Diese Erweiterung erlaubt es mehrere Linien zu zeichnen, die von einem einzigen Punkt ausgehen – die Visualisierung von Abzweigungen ist somit möglich. Die Operatoren **[** und **]** markieren den Anfang und das Ende eines Zweiges.

Beispiel: Mithilfe von verzweigten L-System-Beschreibungen lassen sich simple, Baum- oder Strauch-ähnliche Strukturen bilden. Die Turtle-Interpretation folgt



(a) $n = 0$, $l = 400$, $d = 90°$ (b) $n = 1$, $l = 100$, $d = 90°$

(c) $n = 2$, $l = 25$, $d = 90°$ (d) $n = 3$, $l = 6.25$, $d = 90°$

Abb. 2.4: Die n-fache Ableitung des Axioms $\omega$ anhand der Produktionsregel $p_1$ aus Gleichung 2.6, visualisiert mithilfe der Turtle-Interpretation. Eigene Abbildung.

der in 2.3.1 beschriebenen Interpretation mit fester Strichlänge $l$ und festem Drehwinkel $d$.

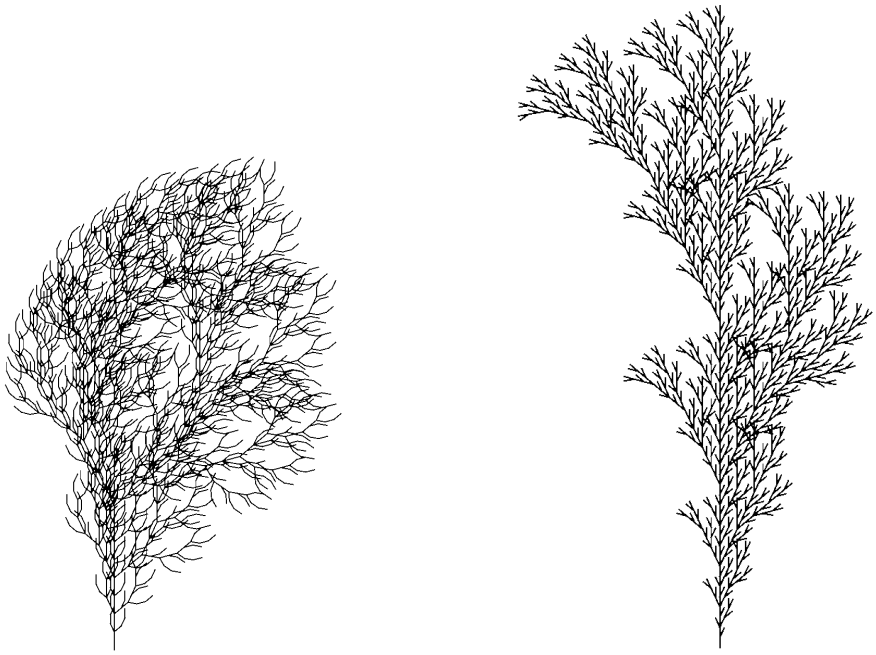
### 2.3.3 Erweiterung der Turtle-Interpretation in den dreidimensionalen Raum

Die bisherige Definition eines Turtle-Zustands mithilfe einer zweidimensionalen Position und einer Blickrichtung genügt nicht, um Visualisierungen von L-Systemen in Form von dreidimensionalen Objekten zu ermöglichen. Sowohl die Zustands-Definition als auch die interpretierten Operationen müssen angepasst und erweitert werden.

Der Zustand der Turtle im dreidimensionalen Raum besitzt eine Position $\overrightarrow{p}$ sowie eine $3 \times 3$ Rotationsmatrix $\boldsymbol{R}$, welche die Orientierung der Turtle im Raum beschreibt. Die Einheitsvektoren $H$, $L$ und $U$ sind orthogonal zueinander und bilden das lokale Koordinatensystem der Turtle.

$\overrightarrow{\boldsymbol{H}}$ Die Blickrichtung (Heading-Vektor) der Turtle. Eine Rotation um diesen Vektor um den Winkel $d$ entspricht der Rotationsmatrix $R_{\overrightarrow{H}}$:

$$R_{\overrightarrow{H}}(d) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(d) & sin(d) \\ 0 & -sin(d) & cos(d) \end{pmatrix} \tag{2.7}$$



(a) $n = 4$, $l = 18$, $d = 25°$

$\omega \ : F$
$p_1 : F \rightarrow FF - [-F + F + F] + [+F - F - F]$
[PL04, S.25]

(b) $n = 5$, $l = 15$, $d = 25°$

$\omega \ : F$
$p_1 : F \rightarrow F[-F]F[+F][F]$
[STN16, S.78]

Abb. 2.5: Die n-fache Ableitung der Axiome anhand der Produktionsregeln, visualisiert mithilfe der Turtle-Interpretation. Eigene Abbildung.

$\overrightarrow{\boldsymbol{L}}$ Der Vektor, der im lokalen Koordinatensystem der Turtle nach links zeigt (Left-Vektor). Eine Rotation um diesen Vektor um den Winkel $d$ entspricht der Rotationsmatrix $R_{\overrightarrow{L}}$:

$$R_{\overrightarrow{L}}(d) = \begin{pmatrix} cos(d) & 0 & sin(d) \\ 0 & 1 & 0 \\ -sin(d) & 0 & cos(d) \end{pmatrix} \tag{2.8}$$

$\overrightarrow{\boldsymbol{U}}$ Der Vektor, der im lokalen Koordinatensystem der Turtle nach oben zeigt (Up-Vektor). Eine Rotation um diesen Vektor um den Winkel $d$ entspricht der Rotationsmatrix $R_{\overrightarrow{U}}$:

$$R_{\overrightarrow{U}}(d) = \begin{pmatrix} cos(d) & -sin(d) & 0 \\ sin(d) & cos(d) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

[PL04, S.19] [DL05, S.69]

Die erweiterte Turtle-Interpretation verarbeitet folgende Symbole:

**$F(l)$** Die Turtle bewegt sich um $l > 0$ in Blickrichtung $\overrightarrow{H}$. Die neue Position ist $\overrightarrow{p_{neu}}$ mit:

$$\overrightarrow{p_{neu}} = \overrightarrow{p} + \boldsymbol{R} * \overrightarrow{H} * l \tag{2.10}$$

Zwischen der alten Position $p$ und der neuen Position $p_{neu}$ wird eine Linie gezeichnet.

**$+(d)$** Die Turtle dreht sich nach links um den Winkel $d$. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{U}}(d) \tag{2.11}$$

**$-(d)$** Die Turtle dreht sich nach rechts um den Winkel $d$. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{U}}(-d) \tag{2.12}$$

**$\&(d)$** Die Turtle neigt sich nach unten um den Winkel $d$. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{L}}(d) \tag{2.13}$$

**$\wedge(d)$** Die Turtle neigt sich nach oben um den Winkel $d$. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{L}}(-d) \tag{2.14}$$

**$\backslash(d)$** Die Turtle rollt sich nach links um den Winkel $d$. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{H}}(d) \tag{2.15}$$

**$/(d)$** Die Turtle rollt sich nach rechts um den Winkel $d$. Die neue Rotationsmatrix entspricht $\boldsymbol{R'}$ mit:

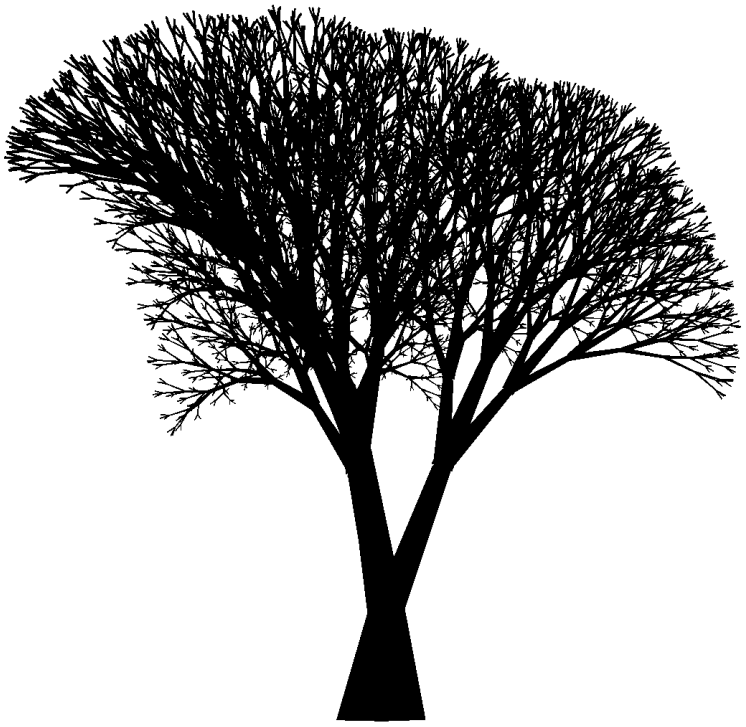
$$\boldsymbol{R'} = \boldsymbol{R} * R_{\overrightarrow{H}}(-d) \tag{2.16}$$

[PL04, S.19] [DL05, S.69]

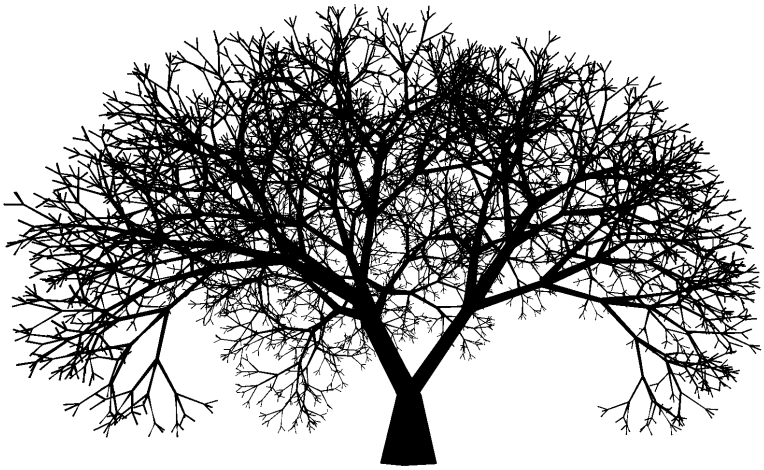
Mithilfe der Turtle-Interpretation im dreidimensionalen Raum können nun L-Systeme visualisiert werden, die Baumstrukturen ähneln. Ein Beispiel dafür ist das L-System aus Gleichung 2.17.

$$\begin{aligned} &\omega : \ /(45)\ A \\ &p_1 : A \quad \rightarrow F(50)\ [\&(a)F(50)A]\ /(d)\ [\&(a)F(50)A]\ /(d)\ [\&(a)F(50)A] \\ &p_2 : F(l) \quad \rightarrow F(l * l_r) \end{aligned} \tag{2.17}$$

[PL04, S.60]



(a) $n = 8$, $d = 137.5°$, $a = 18.95°$, $l_r = 1.3$



(b) $n = 8$, $d = 137.5°$, $a = 36.0°$, $l_r = 1.3$

Abb. 2.6: Die n-fache Ableitung des Axioms anhand der Produktionsregeln aus Gleichung 2.17, visualisiert mithilfe der Turtle-Interpretation. Eigene Abbildung.

# 3

# Space Colonization Algorithmus

TODO: Zusammenfassung Space Colonization Algorithm.

## 3.1 Ursprung

Der Space Colonization Algorithmus wurde ursprünglich zur Modellierung und Visualisierung von Blattvenen vorgeschlagen und basiert auf der Wirkung des Pflanzenhormons Auxin. Dieser Hormonstoff entsteht im Blatt und wird von bereits existierenden Blattvenen angezogen, der resultierende Hormonstrom führt zur Bildung von neuen Venen im Blatt. Die Simulation dieses Vorgangs führt zu realitätsnahen Venenmustern. [RFL+05, Abschn. 2.5]

Mithilfe einer Erweiterung in den dreidimensionalen Raum und Nachbearbeitung der Resultate kann eine Vielfalt von Baum- und Strauchstrukturen generiert werden. [RLP07, Abschn. 1]

## 3.2 Aufbau

Der Algorithmus verarbeitet eine Menge von Einflusspunkten $S$ und baut darauf basierend einen gerichteten Baum-Graphen $G = \langle V, E \rangle$ auf. Folgende Begriffe werden in diesem Zusammenhang verwendet:

***V*** Die Menge der Knotenpunkte. Jeder Knotenpunkt entspricht einem Punkt $\overrightarrow{p_v}$ im dreidimensionalen Raum. [Sch14, S.358]

***E*** Die Menge der Kanten, welche die Vorgänger-Nachfolger-Beziehungen zwischen den Knotenpunkten darstellt. Eine Kante $e \in E$ wird als Tupel $e = (v_1, v_2)$ mit $v_1, v_2 \in V$ dargestellt. Jeder Knoten besitzt maximal einen Vorgänger und eine endliche Menge von Nachfolgern.[Sch14, S.358]

***Wurzel*** Eine Wurzel ist ein Knoten $v \in V$ ohne einen Vorgänger. Der Baum besitzt genau eine Wurzel. [Sch14, S.358]

***Grad*** Der Grad eines Knotens ist definiert als die Anzahl seiner Nachfolger. [Lux14, S.29]

***Tiefe*** Die Tiefe eines Knotens ist definiert als die Länge der Folge von Vorgängern, die durchlaufen werden muss, bis die Wurzel erreicht wurde. Die Wurzel besitzt die Tiefe 0. [Lux14, S.30]

Der Algorithmus benötigt die folgenden Eingaben:

$\boldsymbol{d_i}$ Der Einflussradius. Einflusspunkte prägen den Aufbau des Baums nur, wenn sich Knotenpunkte innerhalb dieses Radius befinden. [RLP07, Abschn. 2]

$\boldsymbol{d_k}$ Der Minimalradius. Befindet sich ein Knotenpunkt innerhalb des Minimalradius um einen Einflusspunkt, wird dieser aus der Menge der Einflusspunkte $S$ entfernt. [RLP07, Abschn. 2]

$\boldsymbol{D}$ Die Schrittweite. Jeder neu generierte Knotenpunkt wird in diesem Abstand zu seinem Vorgänger positioniert. [RLP07, Abschn. 2]

$\overrightarrow{\boldsymbol{T}}$ Der Tropismusvektor. Tropismus ist die Tendenz einer Pflanze in eine bestimmte Richtung zu wachsen, beispielsweise aufgrund einer Lichtquelle oder der Beugung durch Gravitation.[RLP07, Abschn. 3]

## 3.3 Ablauf

Zu Beginn des Algorithmus werden $N$ Einflusspunkte in einem vorgegebenen Bereich generiert. Dieser Einflussbereich signalisiert die Verfügbarkeit von Raum, in dem der Baum wachsen kann. [RLP07, Abschn. 2]

Daraufhin wird der Baum iterativ aufgebaut und durchläuft in jeder Iteration die folgenden Schritte:

**1.** Für jeden Einflusspunkt in $S$ wird der am nächsten liegende Knotenpunkt $v \in V$ bestimmt. Befindet sich $v$ innerhalb des Einflussradius $d_i$ um den Einflusspunkt herum, wird dieser einer zugeordneten Menge $S(v)$ hinzugefügt. $S(v)$ beinhaltet somit alle Einflusspunkte, die einen Einfluss auf den Knotenpunkt ausüben. [RLP07, Abschn. 2]

**2.** Befinden sich Elemente in $S(v)$, wird ein neuer Knotenpunkt $v'$ den Nachfolgern von $v$ hinzugefügt und $v$ als Vorgänger von $v'$ eingetragen. Alle Punkte in $S(v)$ beeinflussen $v'$ in gleichem Maße, die neue Position $\overrightarrow{p_{v'}}$ des Knotenpunkts berechnet sich somit wie folgt:

$$\overrightarrow{p_{v'}} = \overrightarrow{p_v} + D * \tilde{n} \tag{3.1}$$

$$\text{mit } \tilde{n} = \frac{\overrightarrow{n} + \overrightarrow{T}}{\|\overrightarrow{n} + \overrightarrow{T}\|} \tag{3.2}$$

$$\text{und } \overrightarrow{n} = \sum_{s \in S(v)} \frac{\overrightarrow{p_s} - \overrightarrow{p_v}}{\|\overrightarrow{p_s} - \overrightarrow{p_v}\|} \tag{3.3}$$

[RLP07, Abschn. 2]

**3.** Für jeden Einflusspunkt wird überprüft, ob sich ein Knotenpunkt innerhalb des Minimalradius $d_k$ befindet. Existiert ein solcher Knotenpunkt, wird der Einflusspunkt aus der Menge der Einflusspunkte $S$ entfernt. [RLP07, Abschn. 2]

Diese Schritte werden solange ausgeführt, bis alle Einflusspunkte entfernt wurden, sich kein Knotenpunkt im Einflussradius eines Einflusspunktes befindet oder bis eine vorgegebene Maximalanzahl von Iterationen durchgeführt wurde. Abbildung 3.1 zeigt eine beispielhafte Anwendung des Algorithmus.

## 3.4 Modellierung von Baumstrukturen

Der Space Colonization Algorithmus liefert einen Baum-Graphen, der Knotenpunkte enthält, welche Positionen im dreidimensionalen Raum darstellen. Um diese Knotenpunkte in Form von baumähnlichen Strukturen zu visualisieren, wird die Prozedur erweitert. Die Modellierung der Baumstrukturen läuft wie folgt ab:

**1.** Der Einflussbereich wird mit der vorgegeben Anzahl von Einflusspunkten gefüllt. [RLP07, Abschn. 2]

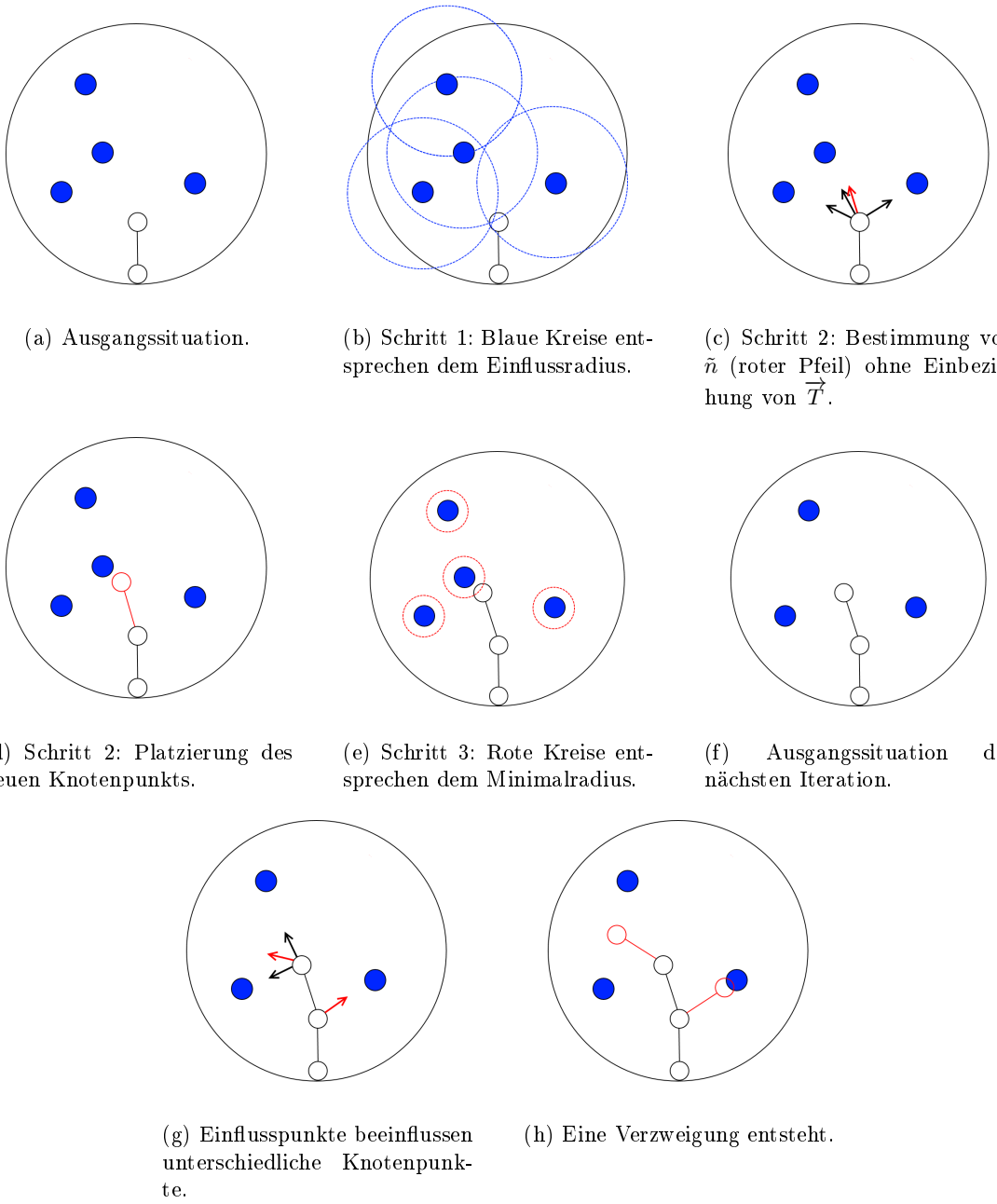
**2.** Der Baum-Graph wird wie in Abschnitt 3.3 beschrieben iterativ generiert. [RLP07, Abschn. 2]

**3.** Die Nachfolger jedes Knotenpunkts werden einander angenähert, um eine Verringerung der Abzweigungswinkel zwischen den verbindenden Kanten zu erreichen. Dies führt zu einer insgesamt realistischeren Baumstruktur. [RLP07, Abschn. 2]

**4.** Die Kanten, welche die Knotenpunkte verbinden, werden mithilfe von Zylindern visualisiert, um die Aststruktur eines Baumes zu simulieren. [RLP07, Abschn. 2]

Die Berechnung der Zylinderbreiten erfolgt anhand von Murrays Regel, die besagt, dass der Radius $r$ eines Astes auf Grundlage der Radien $r_{n_1}...r_{n_m}$ der nachfolgenden, von ihm abzweigenden Äste wie folgt berechnet werden kann:

$$r^g = r^g_{n_1} + r^g_{n_2} + ... + r^g_{n_m} \tag{3.4}$$



(a) Ausgangssituation.

(b) Schritt 1: Blaue Kreise entsprechen dem Einflussradius.

(c) Schritt 2: Bestimmung von $\tilde{n}$ (roter Pfeil) ohne Einbeziehung von $\overrightarrow{T}$.

(d) Schritt 2: Platzierung des neuen Knotenpunkts.

(e) Schritt 3: Rote Kreise entsprechen dem Minimalradius.

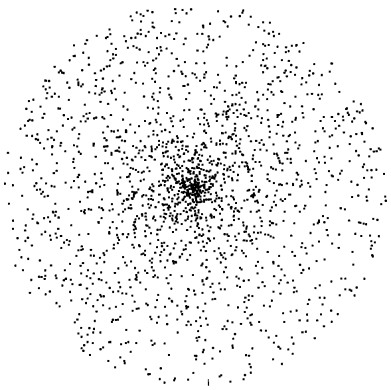
(f) Ausgangssituation der nächsten Iteration.

(g) Einflusspunkte beeinflussen unterschiedliche Knotenpunkte.

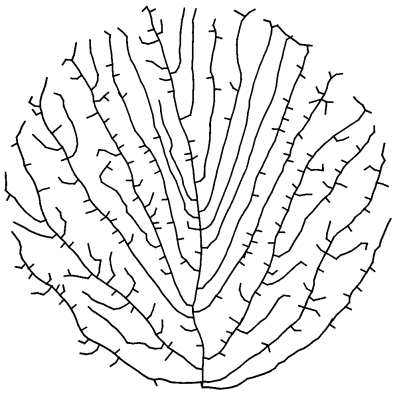
(h) Eine Verzweigung entsteht.

Abb. 3.1: Beispielhafte Anwendung des Space Colonization Algorithmus. Blaue Punkte entsprechen Einflusspunkten, weiße Punkte entsprechen Knotenpunkten. Der unterste Knotenpunkt stellt die Wurzel dar. Eigene Abbildungen auf Grundlage von [RLP07, Abb. 2].

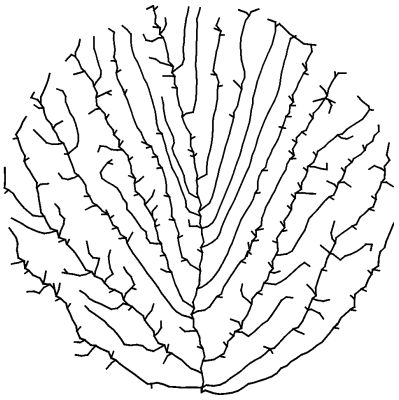
Diese Berechnung kann rekursiv auf dem Baum-Graphen ausgeführt werden, indem für jeden Knotenpunkt der Radius aus den Radien seiner Nachfolger berechnet wird. Besitzt ein Knotenpunkt keine Nachfolger, wird ein festgelegter Radius $r_0$ zurückgegeben. [RFL+05, Abschn. 3.5] Der Wert $g$ kann frei gewählt werden.



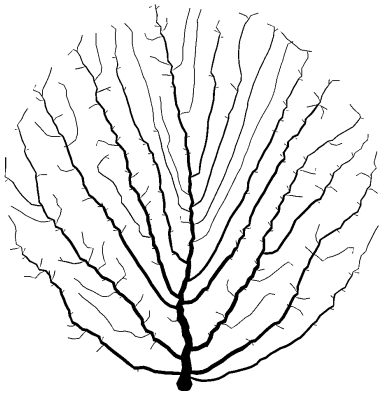
(a) $N = 2000$ Einflusspunkte, zufällig in einem Ring mit dem Radius $r = 500$ verteilt.



(b) Ergebnis des Space Colonization Algorithmus mit einem Einflussradius $d_i = 100$, Minimalradius $d_k = 20$, Schritteweite $D = 15$ und $\overrightarrow{T} = \overrightarrow{0}$.



(c) Verringerung der Abzweigungswinkel.



(d) Modellierung des Ergebnis mithilfe von Zylindern mit $r_0 = 1$ und $g = 2$.

Abb. 3.2: Modellierung einer zweidimensionalen Baumstruktur entsprechend der in Abschnitt 3.4 beschriebenen Schritte. Eigene Abbildungen.

Die von Runions u.a. [RLP07] vorgeschlagene Kurven-Unterteilung [RLP07, Abschn. 2] wurde in dieser Arbeit nicht behandelt, da durch Angabe der Schrittweite $D$ eine ausreichende visuelle Qualität erzielt werden kann.

Abbildung 3.2 zeigt die Modellierung einer zweidimensionalen Baumstruktur.

## 3.5 Erweiterungen

Die Zweigtiefe $Z(v)$ eines Knotens $v \in V$ wird für die Implementierung und Erweiterung des Space Colonization Algorithmus verwendet. Sie entspricht nicht der

Tiefe des Knotens und berechnet sich wie folgt:

$$Z(v) = \begin{cases} 0 & \text{falls v die Wurzel ist} \\ Z(v') & \text{falls v' genau einen Nachfolger besitzt} \\ 1 + Z(v') & \text{sonst} \end{cases} \tag{3.5}$$

wobei $v'$ den Vorgänger des Knotens $v$ darstellt.

Um mehr Kontrolle über die vom Space Colonization Algorithmus generierte Baumstruktur zu bekommen, wurden zusätzliche Bedingungen für die Erstellung von neuen Knotenpunkten hinzugefügt:

***max*$_{grad}$** Der maximale Grad der Knotenpunkte im Baum-Graphen. Bevor ein neuer Knotenpunkt $v_n$ als Nachfolger eines Knotenpunkts $v$ erstellt wird, wird die Anzahl der Nachfolger von $v$ überprüft. Entspricht diese $max_{grad}$, wird der Knotenpunkt $v_n$ nicht erstellt.

***max*$_Z$** Die maximale Zweigtiefe eines Knotens. Bevor ein neuer Knotenpunkt $v'$ als Nachfolger eines Knotenpunkts $v$ erstellt wird, wird seine Zweigtiefe $Z(v')$ berechnet. Übersteigt diese $max_Z$, wird der Knotenpunkt $v'$ nicht erstellt.

Weiterhin wurde gewichtetes Wachstum eingeführt – eine Möglichkeit die Schrittweite $D$ in Abhängigkeit von der Zweigtiefe zu erhöhen. Befindet sich ein Knotenpunkt auf der Zweigtiefe 0, werden neu hinzugefügte Nachfolger im Abstand von $2 * D$ positioniert, befindet er sich auf der maximalen Zweigtiefe, werden neu hinzugefügte Nachfolger im Abstand von $D$ positioniert. Zwischen diesen beiden Zweigtiefen wird die Schrittweite linear interpoliert.

Kurvenreduktion in Abhängigkeit des Abzweigungswinkel ermöglicht es die Baumstruktur mithilfe einer verringerten Datenmenge darzustellen. Besitzt ein Knoten $v$ genau einen Nachfolger $v_n$ und einen Vorgänger $v'$, werden die zwei Richtungsvektoren zwischen $v'$ und $v$ sowie $v$ und $v_n$ berechnet:

$$\overrightarrow{R_1} = \frac{p_v - p_{v'}}{\|p_v - p_{v'}\|} \text{ und } \overrightarrow{R_2} = \frac{p_{v_n} - p_v}{\|p_{v_n} - p_v\|} \tag{3.6}$$

Übersteigt das Skalarprodukt $\langle \overrightarrow{R_1}, \overrightarrow{R_2} \rangle$ einen festgelegten Maximalwert, wird $v$ aus dem Baum entfernt. $v_n$ wird zum Nachfolger von $v'$ und $v'$ zum Vorgänger von $v_n$.

# 4

# Implementierung

## 4.1 Baumstruktur

## 4.2 L-Systeme

### 4.2.1 L-System-Plant

### 4.2.2 Turtle-Graphic-Interpreter

### 4.2.3 Performanz

## 4.3 Space-Colonization-Algorithmus

### 4.3.1 Colonization Space

### 4.3.2 Space-Colonization-Plant

## 4.4 Mesh-Generierung

# 5

# Ergebnisse und Vergleich

## 5.1 Visuell

## 5.2 Performanz

## 5.3 Probleme

Mit SCA: Gleicher Abstand von zwei Einflusspunkten führt zu beinahe unendlicher Hin- und Her-Generierung von Branches.

# 6

# Zusammenfassung und Ausblick

Zusammenfassung

## 6.1 Erweiterungen

### 6.1.1 Aststruktur

Generalized cylinders.

### 6.1.2 Texturen

### 6.1.3 Blätter

### 6.1.4 Generierung zur Laufzeit

### 6.1.5 Verteilung

# Literaturverzeichnis


DL05. Deussen, Oliver und Bernd Lintermann: *Digital Design of Nature - Computer Generated Plants and Organics*. Springer-Verlag Berlin Heidelberg 2005, 2005.

GSJ04. Goldman, Ron, Scott Schaefer und Tao Ju: *Turtle Geometry in Computer Graphics and Computer Aided Design*, 2004. `http://www.cs.wustl.edu/~taoju/research/TurtlesforCADRevised.pdf`.

Lux14. Lux, Prof. Dr. Andreas: *Algorithmen und Datenstrukturen - Vorlesungsskript Kapitel 4*, 2014.

Man83. Mandelbrot, Benoit B.: *The Fractal Geometry of Nature*. W. H. Freeman and Company, 1983.

PL04. Prusinkiewicz, Przemyslaw und Aristid Lindenmayer: *The Algorithmic Beauty of Plants*. Przemyslaw Prusinkiewicz, eBook Auflage, 2004. `http://algorithmicbotany.org/papers/abop/abop.pdf`.

RFL+05. Runions, Adam, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan und Przemyslaw Prusinkiewicz: *Modeling and visualization of leaf venation patterns*, 2005. `http://algorithmicbotany.org/papers/venation.sig2005.pdf`.

RLP07. Runions, Adam, Brendan Lane und Przemyslaw Prusinkiewicz: *Modeling Trees with a Space Colonization Algorithm*, 2007. `http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf`.

Sch14. Schmitz, Prof. Dr. Heinz: *Theoretische Informatik - Vorlesungsskript*, 2014.

STN16. Shaker, Noor, Julian Togelius und Mark J. Nelson: *Procedural Content Generation in Games*. Springer International Publishing Switzerland 2016, 2016.

# A

# Glossar

# B

## Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist ...

Diesen Teil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser: ...

| | |
|---|---|
| Datum | Unterschrift der Kandidatin / des Kandidaten |